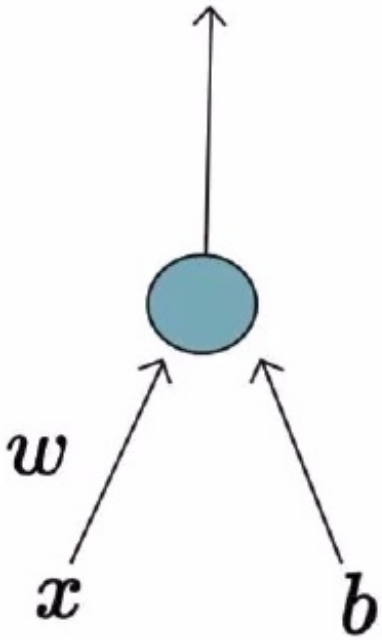
Back Propagation.

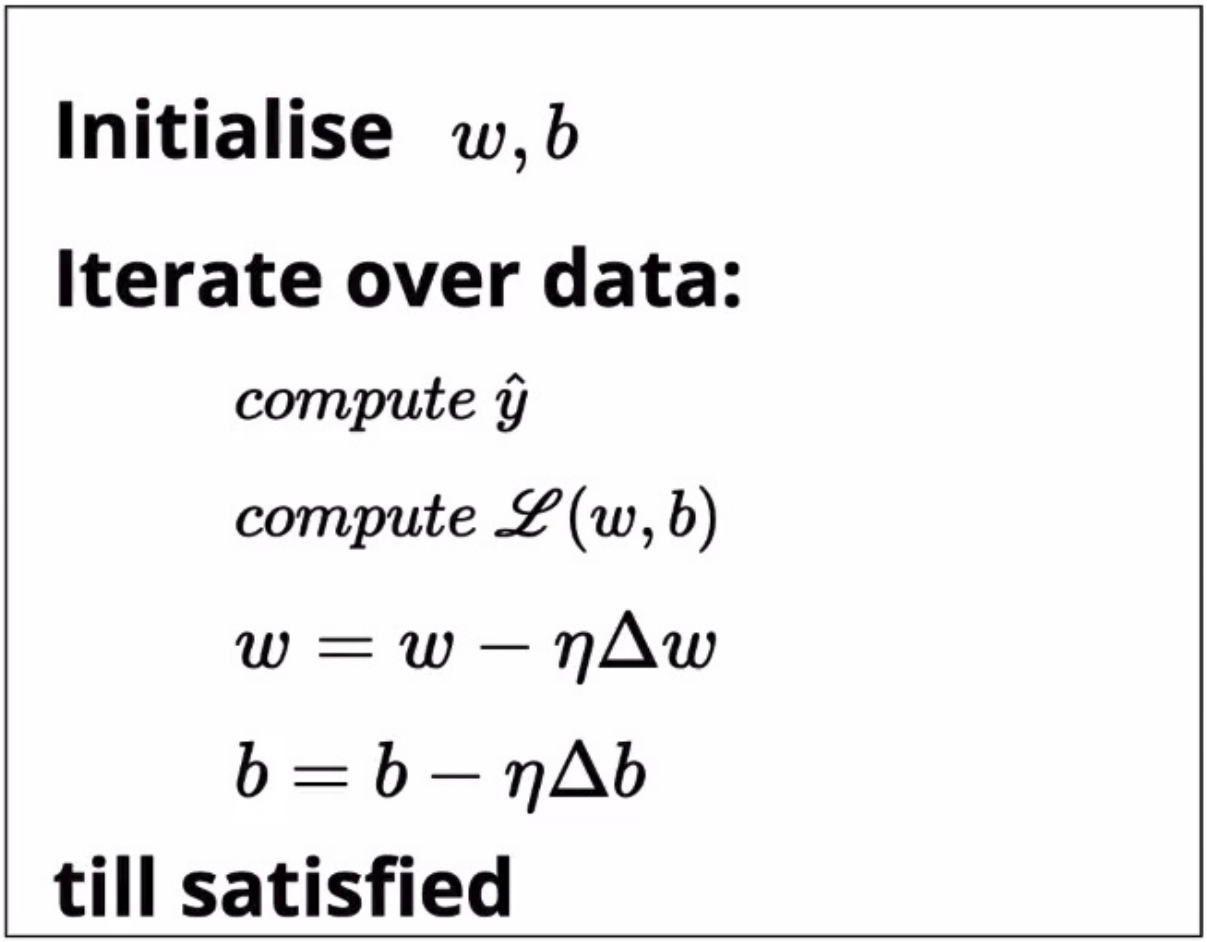
medium.com/@manveetdn/back-propagation-15c02aba112

The Vectorised / Heavy Math Part.

> **Disclaimer: This is notes on "Back Propagation(Vectorised/Heavy-Maths part)" Lesson (PadhAI onefourthlabs course "A First Course on Deep Learning")**

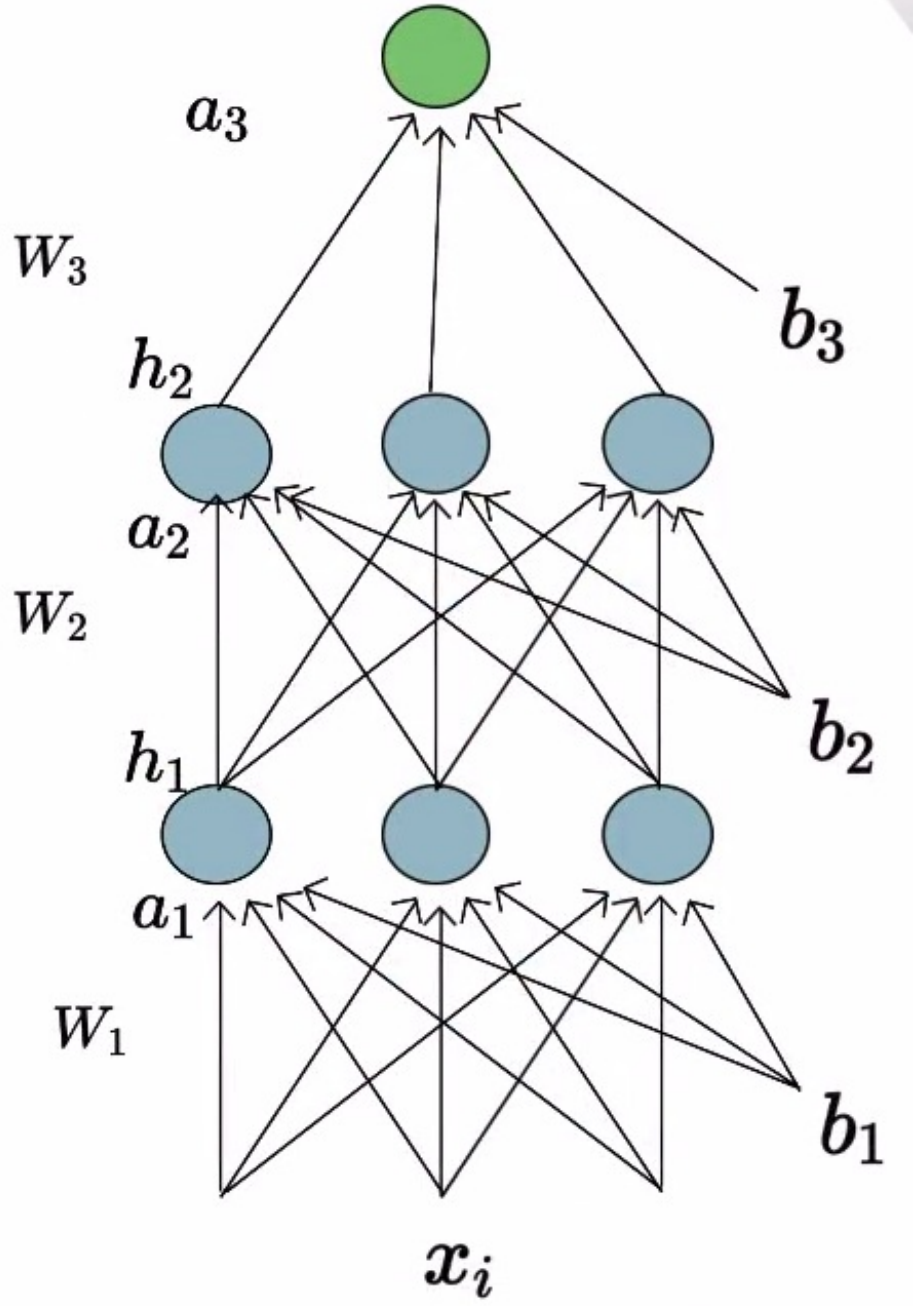
Lets start with simple basic Neuron and the Learning algorithm for that as Shown below.





Basic Neuron and the Learning Algorithm for that.

Using this we will build a Deep neural Network, Accordingly the Learning Algorithm for that Network as before we know.


a_3
W_3
b_3
h_2
a_2
W_2
b_2
h_1
a_1
W_1
b_1
x_i

Initialise w, b

Iterate over data:

$compute\ \hat{y}$

$compute\ \mathscr{L}(w, b)$

$$w_{111} = w_{111} - \eta \Delta w_{111}$$

$$w_{112} = w_{112} - \eta \Delta w_{112}$$

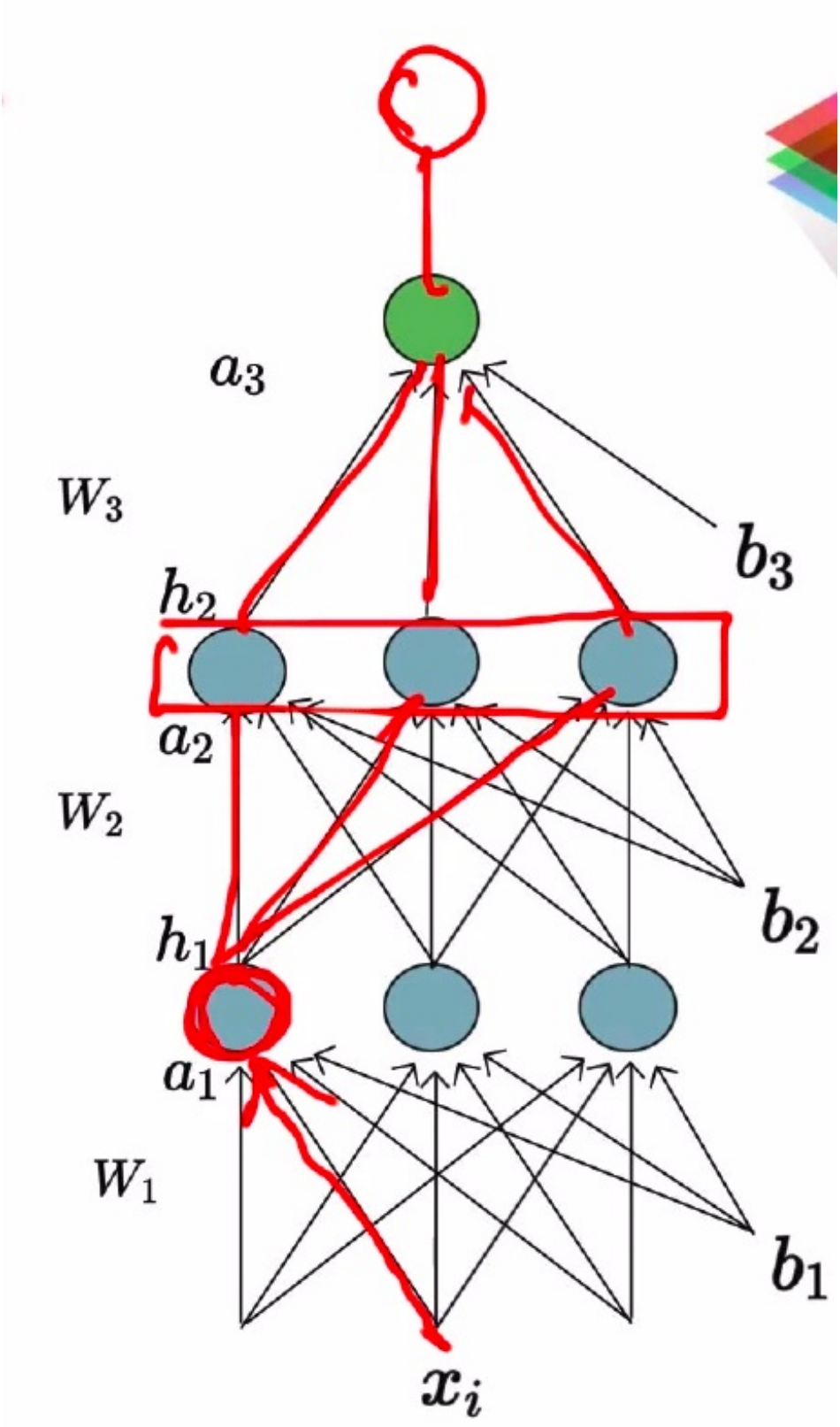
....

$$w_{313} = w_{313} - \eta \Delta w_{313}$$

till satisfied

This the complex Neural Network ans we these beside is the respective algorithm for that.

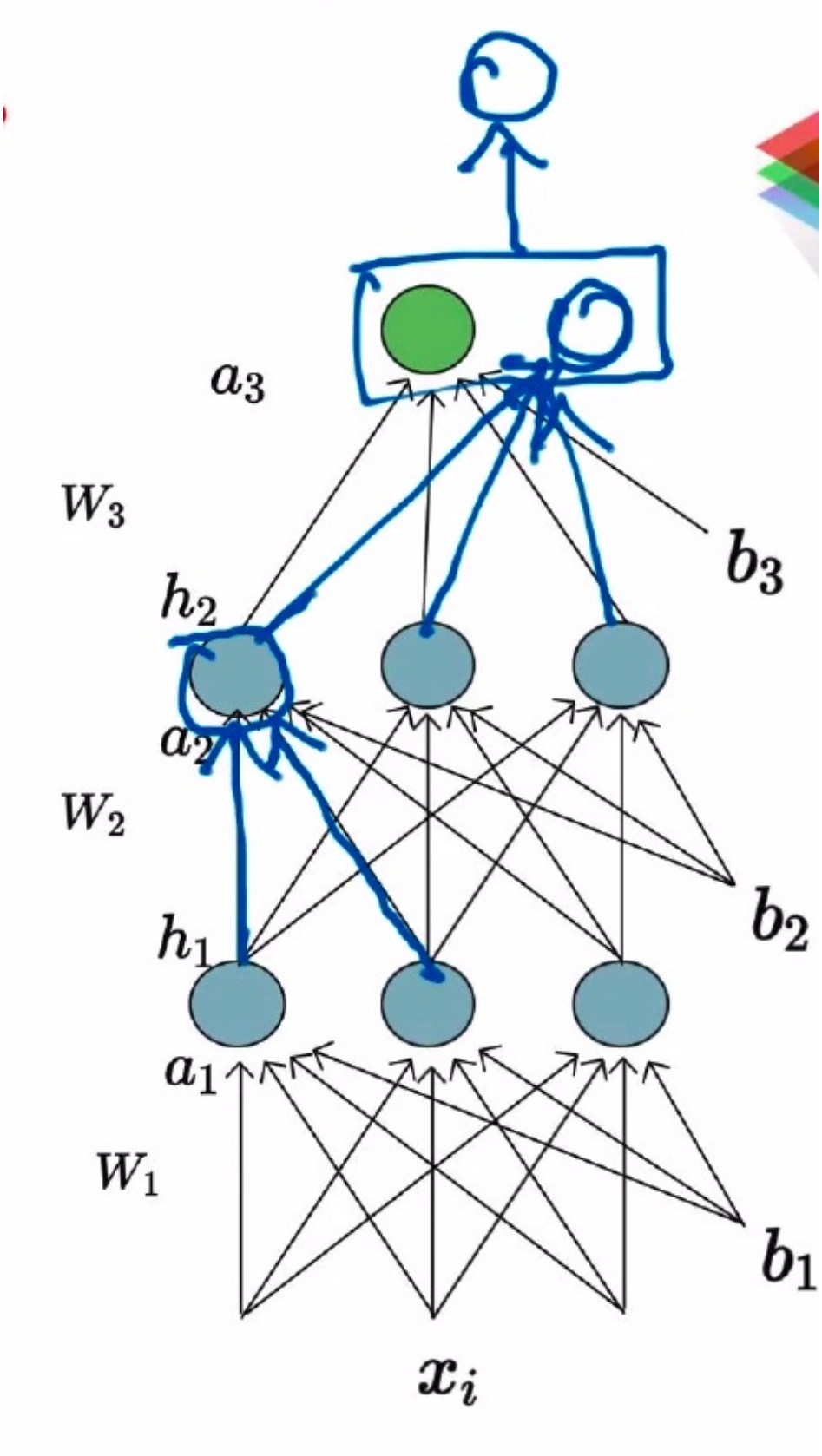
Using the Gradient Descent rule we will update each weight as shown in the above two in the learning the Algorithm part.



Taking the neuron a1 in the first layer.

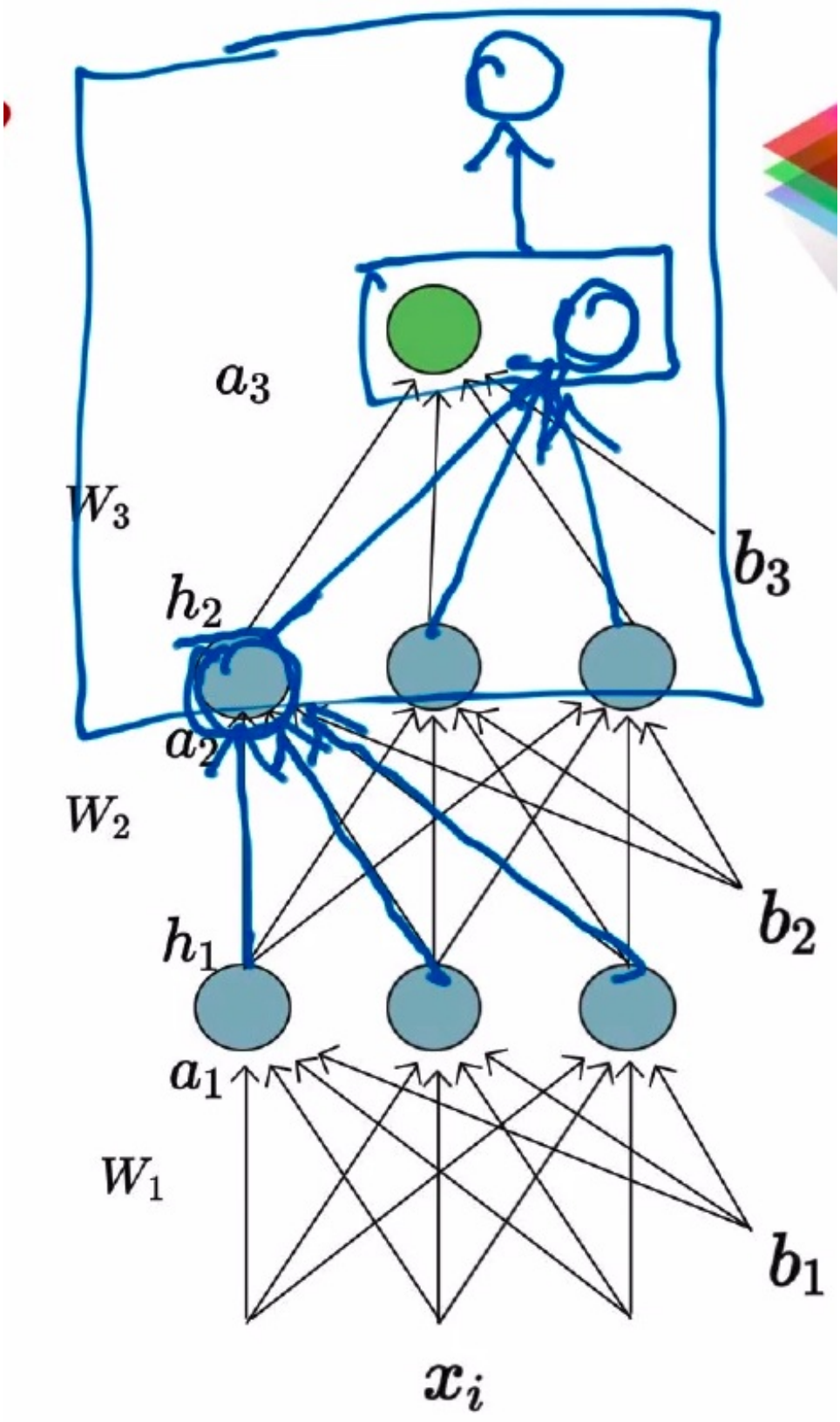
Here, we shall **concentrate on a1** and it is connected to every other neurons in the second layer and marked with red a highlighted by drawing a box around them and "**x ᵢ"** weight with which it is multiplies will effect that neuron and that effects the next three neuron in the above layer and which surely **influences a change in the output**.

Let’s take the other situation where here, we will consider another neuron in the third layer and the final output is dependent on the two neurons in the third layer as shown below.



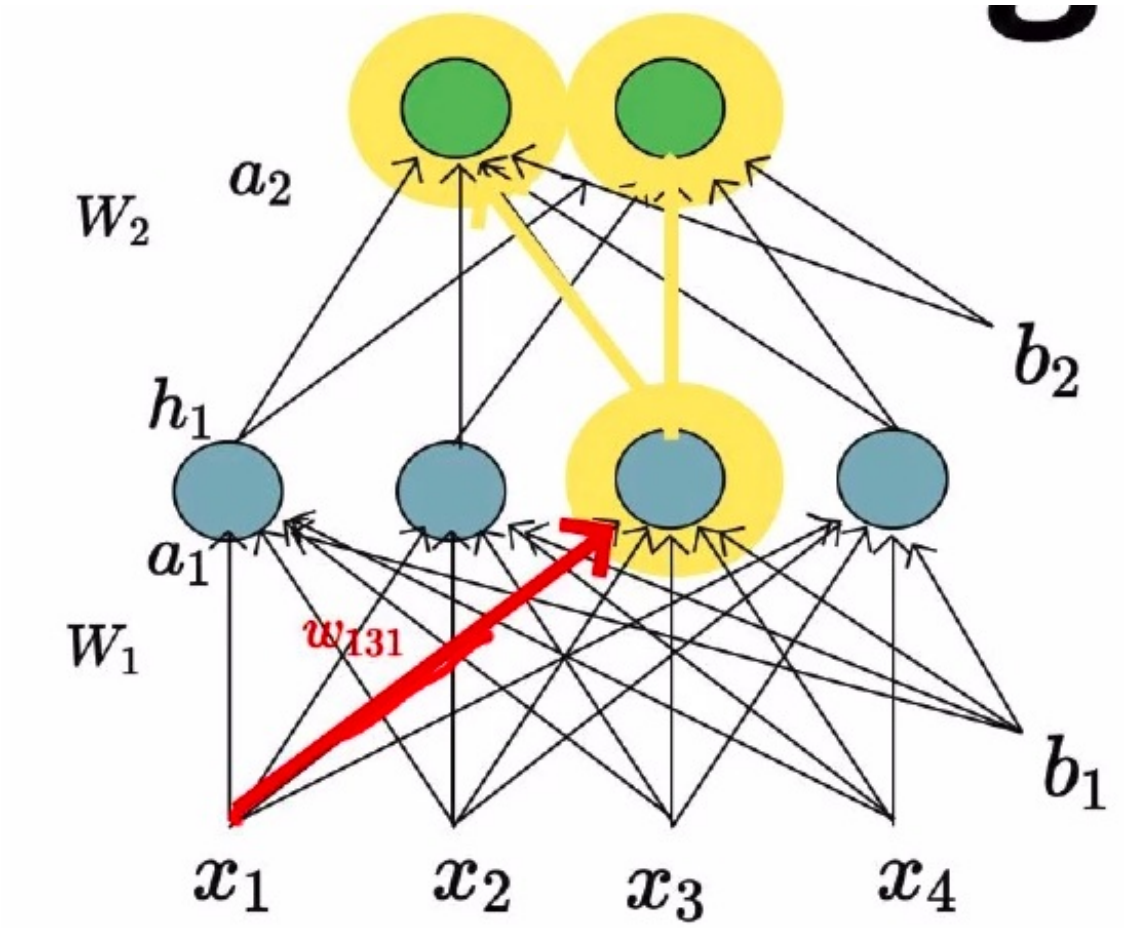
if we consider another neuron in the third layer and then the final then outputs depends on the two neuron in 3rd layer now.

Here, the new neuron is connected to all the neurons of second layer as shown and the these two in the third layer are connected to the loss function. so to compute loss of any neuron from here we are gonna back propagate through the seconds and third layer remain the same. for any weight we wanna change.Its is squared with rectangle in the below figure.



Then, like that part will common example here we take the the circled neuron a2 in he second layer then if we wanna back propagate the we need to take all the paths backward as shown blow and it influence all the neuron in the above layers after updating. That's the idea

behind back propagation. Computing the multiple weights at one go.



Here, In this particular network we are going to stress on the w_{131} then we the loss function derivative is given.

$$\frac{\partial L}{\partial w_{131}} = \left(\frac{\partial L}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial a_{21}} \cdot \frac{\partial a_{21}}{\partial h_{13}} + \frac{\partial L}{\partial \hat{y}_2} \cdot \frac{\partial \hat{y}_2}{\partial a_{22}} \cdot \frac{\partial a_{22}}{\partial h_{13}}\right).\left(\frac{\partial h_{13}}{\partial a_{13}}\right).\left(\frac{\partial a_{13}}{\partial w_{131}}\right)$$

$$\frac{\partial L}{\partial \hat{y}_1} = -2(y_1 - \hat{y}_1)$$

This is the directly written for the derivative of Loss function w.r.t w_{131}. If we take the first formulae we can reuse the whole thing except the last term

($\partial a_{13}/\partial w_{131}$) will only change the remaining all terms can be reused.

Then firstly we will starts with the derivative of the gradient withe the derivative of the a_2 neuronin the first layer. which is represented as ($\partial L/\partial a_2$)

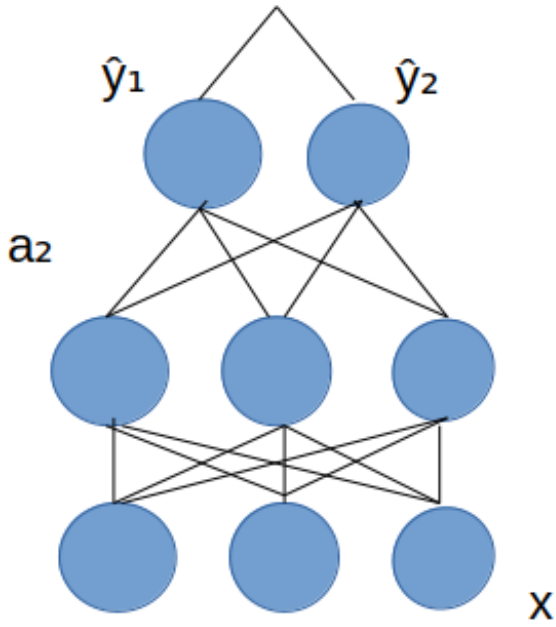
Which will the matrix of the derivative of loss function w.r.t a_{21} and a_{22}.

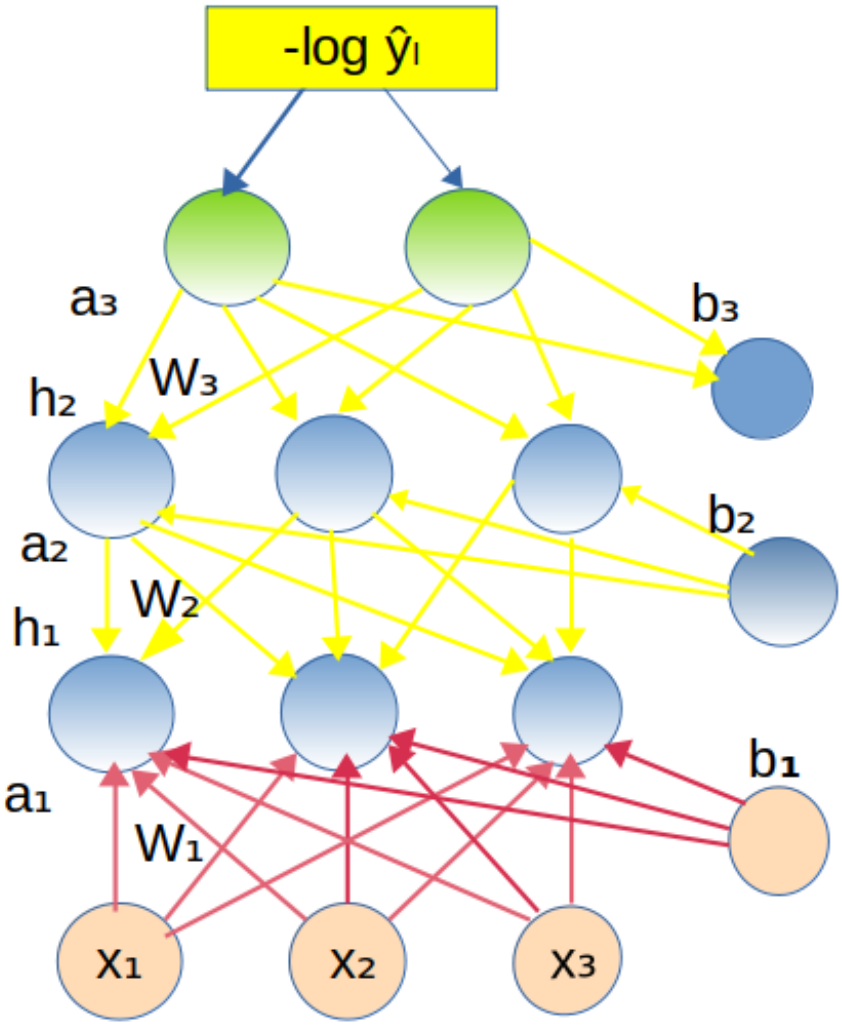
Let us see an intuitive explanation of back propagation before we get into the mathematical part.

$\hat{y} = [\hat{y}_1\ \hat{y}_2]$

$a_2 = [a_{21}\ a_{22}]$

like this the **$\hat{y}$** and **a_2 defined** and beside is the basic network





Here we consider the following neural network then after calculating the loss with help of loss function if we get any loss then we will go to the third layer and then we will know that the third layer takes the values given from the previous layer and applies a soft max function.Then we will go to the second layer then here w_3 ,b_3 are the weights and again **h2 says that its is from the weights are borrowed from the first layer** where w_2,b_2 are the weights and like that we will back propagate until the weights and then finally we come to the inputs and then we will change the w_1,b_1 in the first layer and like that we update all weight and reduce the loss function values to the maximum possible extent.

$$f(x) = \hat{y} = O(W_L h_{L-1} + b_L)$$

Formula format of the above one process.

w_1,b_1,w_2,b_2,w_3,b_3 values play a key role in this the loss function, Updating their values plays a key role in the final loss value.How to update them and what is the procedure need to be followed is given by the gradient descent (We need to move in the opposite direction of the gradient).

$$\frac{\partial L(\theta)}{\partial W_{111}} = \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3} \frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2} \frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1} \frac{\partial a_3}{\partial W_{111}}$$

Going tthe output layer calculating the dervative. | Talk to the ouput layer | Talk to the previous hidden layer | Talk to the previous hidden layer | Now talk to the weight

Communication through Hidden layers see the description below to get a more clear understanding of this.

> **In more detailed way**

> **We can say that W_L, b_L and h_L and then ask them "What is wrong with you".**

> **W_L and b_L take full responsibility but h_L says"Well please understand that I am only as good as the pre-activation Layer"**

> **The pre-activation layer in turn says that I am only as good as the hidden layer and the weights below me.**

> **We continue in that manner and realise that the responsibility lies with all the weights and biases(i.e., parameters of the model).**

> **But instead of talking to them directly , it is easier to talk to them through the hidden layer and the output layers(and this is exactly the chain rule allows us to do).**

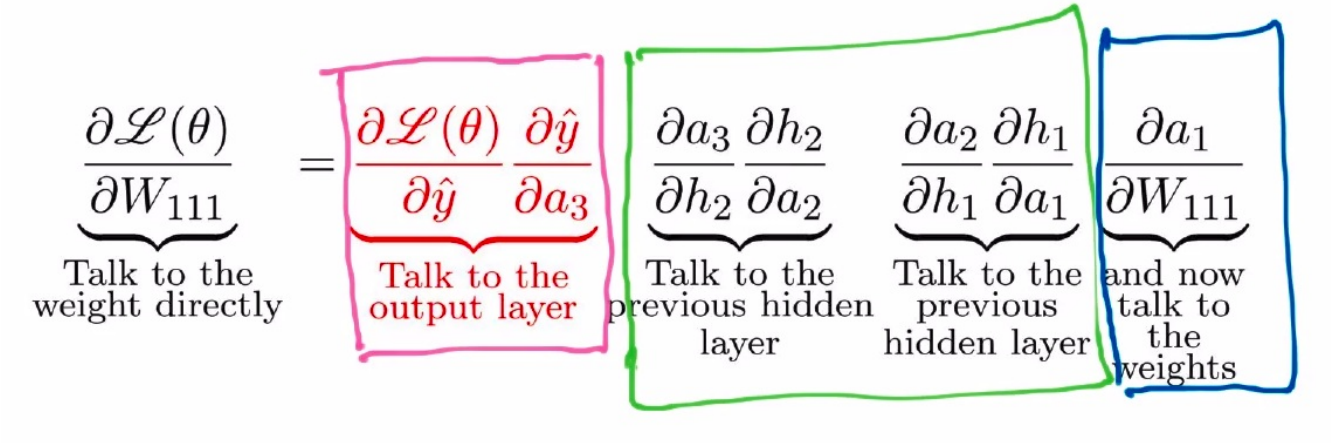
Quantities of interest:

We will divide the further computation in 3 parts namely.(Focus is on Cross entropy and the Softmax output.

1.Gradient w.r.t output units

2.Gradient w.r.t hidden units

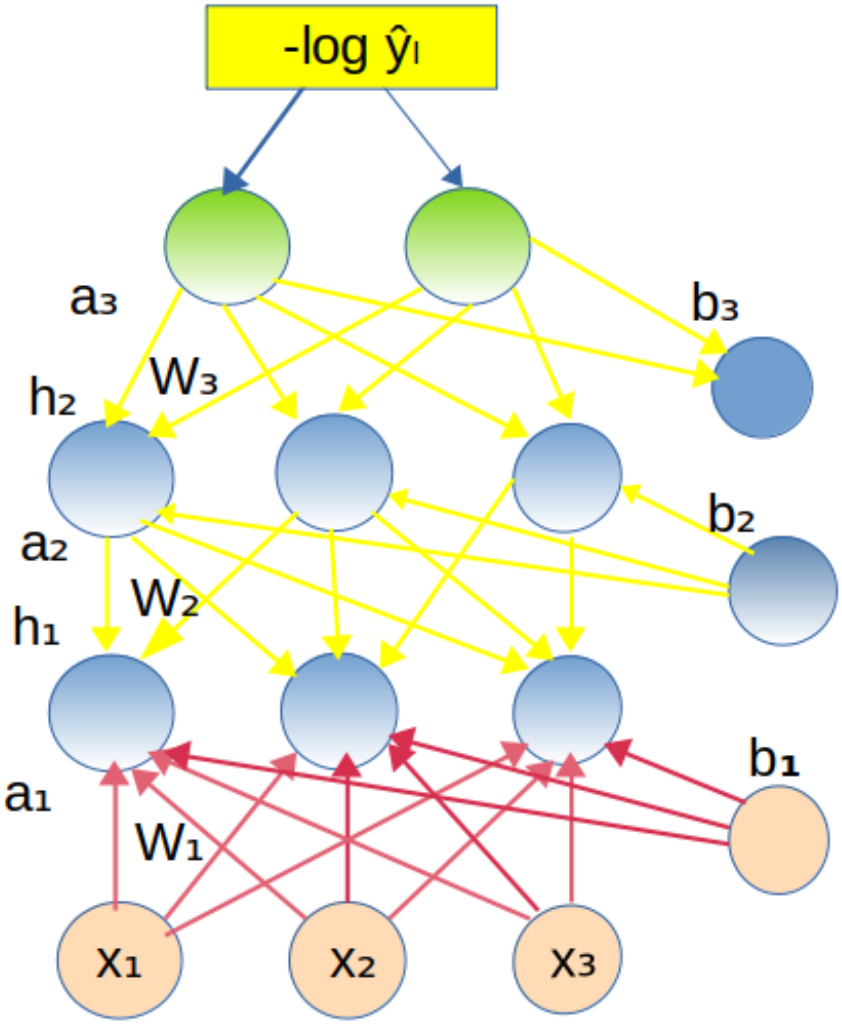
3.Gradient w.r.t weights



dividing into 3 parts.

Lets consider the first part that is the highlighted with red in the given formulae above.

$$\frac{\partial \mathscr{L}(\theta)}{\partial a_{Li}} = \frac{\partial(-\log \hat{y}_\ell)}{\partial a_{Li}}$$



Lets consider the a_3 in the figure beside figure here we are mostly taking the above formulae here **L is 3 and the i ranges from {1,2}** because there are only 2 vertices(i.e. neurons) like that if it has **k vertices or neuron it ranges from {1,2,.....k}**in this level We will be using like that the above formulae. like that we calculate the derivative independently for each and every neuron.

For finding this we will derive formula for neuron with vertices(i,j,k) and we will generalise that formula to the entire layer.

In this case of hidden also we will get **i has the layer number** and **j as the neuron number**.Then we can calculate the derivative of the loss function w.r.t to any neuron in the hidden layer.

Gradient w.r.t output units:

What we are actually interested in is

$$\frac{\partial \mathscr{L}(\theta)}{\partial a_{Li}} = \frac{\partial(-\log \hat{y}_\ell)}{\partial a_{Li}}$$
$$= \frac{\partial(-\log \hat{y}_\ell)}{\partial \hat{y}_\ell} \frac{\partial \hat{y}_\ell}{\partial a_{Li}}$$

Does $\hat{y}_\ell$ depend on a_{Li} ? Indeed, it does.

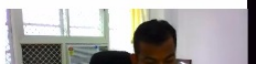
$$\hat{y}_\ell = \frac{exp(a_{L\ell})}{\sum_i exp(a_{Li})}$$

Having established this, we will now derive the full expression on the next slide

We will cut the loss function down into smaller parts and then we find the formula for the individual neuron.

$$\frac{\partial}{\partial a_{Li}} - \log \hat{y}_\ell = \frac{-1}{\hat{y}_\ell} \frac{\partial}{\partial a_{Li}} \hat{y}_\ell$$
$$= \frac{-1}{\hat{y}_\ell} \frac{\partial}{\partial a_{Li}} softmax(\mathbf{a}_L)_\ell$$
$$= \frac{-1}{\hat{y}_\ell} \frac{\partial}{\partial a_{Li}} \frac{\exp(\mathbf{a}_L)_\ell}{\sum_{i'} \exp(\mathbf{a}_L)_\ell}$$
$$= \frac{-1}{\hat{y}_\ell} \left(\frac{\frac{\partial}{\partial a_{Li}} \exp(\mathbf{a}_L)_\ell}{\sum_{i'} \exp(\mathbf{a}_L)_{i'}} - \frac{\exp(\mathbf{a}_L)_\ell \left(\frac{\partial}{\partial a_{Li}} \sum_{i'} \exp(\mathbf{a}_L)_{i'} \right)}{(\sum_{i'} (\exp(\mathbf{a}_L)_{i'})^2} \right)$$
$$= \frac{-1}{\hat{y}_\ell} \left(\frac{\mathbb{1}_{(\ell=i)} \exp(\mathbf{a}_L)_\ell}{\sum_{i'} \exp(\mathbf{a}_L)_{i'}} - \frac{\exp(\mathbf{a}_L)_\ell}{\sum_{i'} \exp(\mathbf{a}_L)_{i'}} \frac{\exp(\mathbf{a}_L)_i}{\sum_{i'} \exp(\mathbf{a}_L)_{i'}} \right)$$
$$= \frac{-1}{\hat{y}_\ell} \left(\mathbb{1}_{(\ell=i)} softmax(\mathbf{a}_L)_\ell - softmax(\mathbf{a}_L)_\ell softmax(\mathbf{a}_L)_i \right)$$
$$= \frac{-1}{\hat{y}_\ell} \left(\mathbb{1}_{(\ell=i)} \hat{y}_\ell - \hat{y}_\ell \hat{y}_i \right)$$
$$= -\left(\mathbb{1}_{(\ell=i)} - \hat{y}_i \right)$$

$$\frac{\partial \frac{g(x)}{h(x)}}{\partial x} = \frac{\partial g(x)}{\partial x} \frac{1}{h(x)} - \frac{g(x)}{h(x)^2} \frac{\partial h(x)}{\partial x}$$



Here, we consider the expression and we will simplify that accordingly we will use the **derivative(u/v)** formula and like that we will simplify accordingly and we will come to a conclusion.

$$\frac{\partial \mathscr{L}(\theta)}{\partial a_{L,i}} = -\left(\mathbb{1}_{\ell=i} - \hat{y}_i \right)$$

Finally, we have come t a conclusion and then we now we will compute the gradient w.r.t aL that is as follows.

So far we have derived the partial derivative w.r.t. the i-th element of $\mathbf{a}_L$

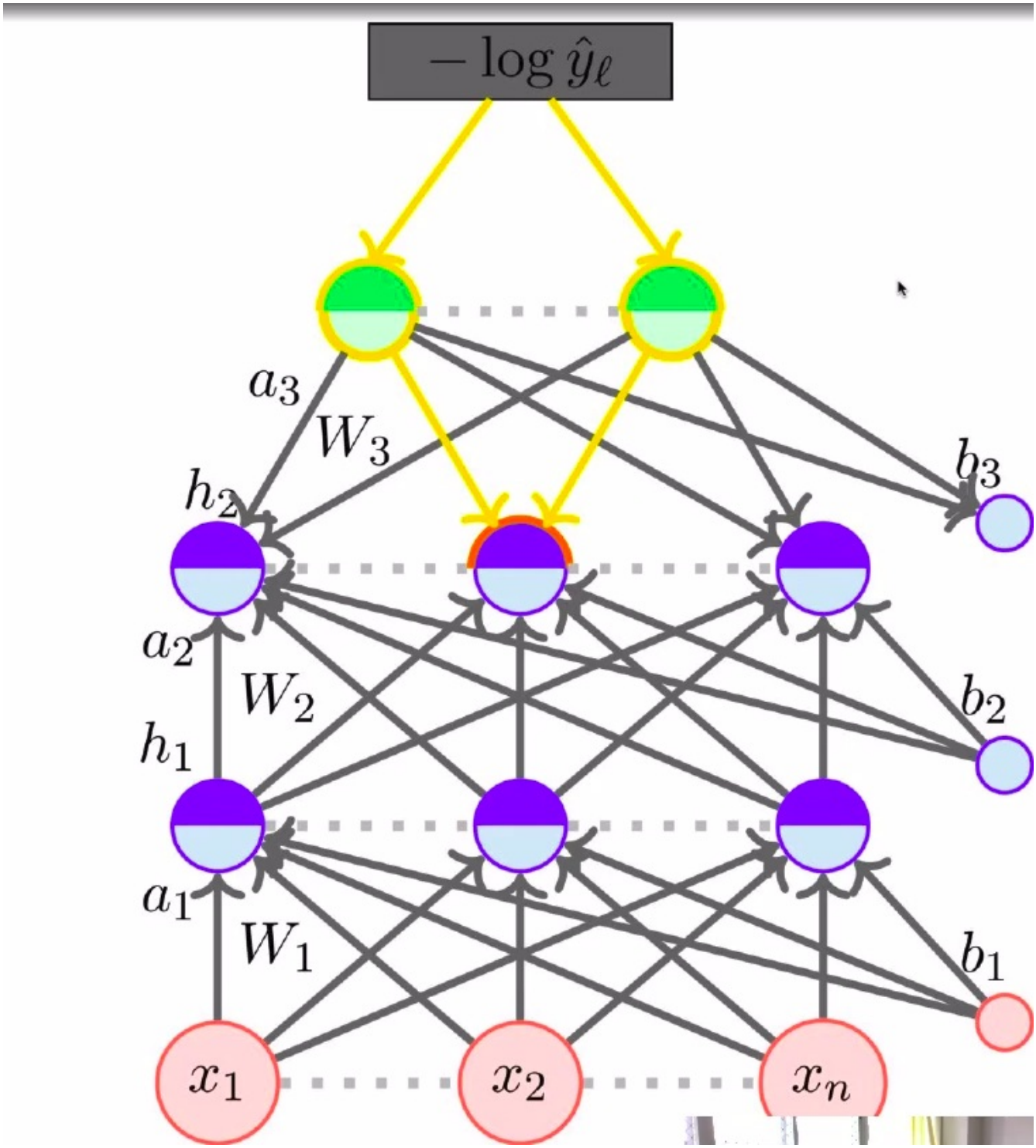
$$\frac{\partial \mathscr{L}(\theta)}{\partial a_{L,i}} = -(\mathbb{1}_{\ell=i} - \hat{y}_i)$$

We can now write the gradient w.r.t. the vector $\mathbf{a}_L$

$$\nabla_{\mathbf{a_L}} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial a_{L1}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial a_{Lk}} \end{bmatrix} = \begin{bmatrix} -(\mathbb{1}_{\ell=1} - \hat{y}_1) \\ -(\mathbb{1}_{\ell=2} - \hat{y}_2) \\ \vdots \\ -(\mathbb{1}_{\ell=k} - \hat{y}_k) \end{bmatrix}$$
$$= -(\mathbf{e}(\ell) - \hat{y})$$

Finally, calculating the gradient we arrive at the conclusion that gradient is - **e(L)- ŷ)** which is **- (y- ŷ)** Which is the negative of the difference between the true value and the predicted value.

Gradient w.r.t hidden units:



Here, we consider the dark blue part of the neuron hidden layer that is the second layer from the top again we already have the the light green portions before now we are going to compute the dark blue portions i.e., the derivative of the loss function w.r.t to this parts.

$$\frac{\partial \mathscr{L}(\theta)}{\partial h_{ij}} = \sum_{m=1}^{k} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i+1,m}} \frac{\partial a_{i+1,m}}{\partial h_{ij}}$$

$$= \sum_{m=1}^{k} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i+1,m}} W_{i+1,m,j}$$

Now consider these two vectors,

$$\nabla_{a_{i+1}} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i+1,1}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial a_{i+1,k}} \end{bmatrix} ; W_{i+1,\cdot,j} = \begin{bmatrix} W_{i+1,1,j} \\ \vdots \\ W_{i+1,k,j} \end{bmatrix}$$

$W_{i+1,\cdot,j}$ is the j-th column of W_{i+1}; see that,

$$(W_{i+1,\cdot,j})^T \nabla_{a_{i+1}} \mathscr{L}(\theta) = \sum_{m=1}^{k} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i+1,m}} W_{i+1,m,j}$$

From all the computation we get the conclusion formula of the dot product of the weights which is.

$$(W_{i+1,\cdot,j})^T \nabla_{a_{i+1}} \mathscr{L}(\theta) = \sum_{m=1}^{k} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i+1,m}} W_{i+1,m,j}$$

W already know that the right side part of the formulae is derivative of loss function w.r.t the **h_{ij} .**

$$\frac{\partial \mathscr{L}(\theta)}{\partial h_{ij}} = \sum_{m=1}^{k} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i+1,m}} \frac{\partial a_{i+1,m}}{\partial h_{ij}}$$

Now we can say the the derivative is the of the loss function w.r.t w is the dot product of the weights we assigned.

$$, \frac{\partial \mathscr{L}(\theta)}{\partial h_{ij}} = (W_{i+1,.,j})^T \nabla_{a_{i+1}} \mathscr{L}(\theta)$$

The above formulae is fro one independent neuron in the particular layer and using this we need to compute by taking each and every neuron one by one which is again a complex thing so to overcome that, we will derive a formula with which we can compute all the gradient of the all the neurons at once in particular layer.

$$We\ have, \frac{\partial \mathscr{L}(\theta)}{\partial h_{ij}} = (W_{i+1,.,j})^T \nabla_{a_{i+1}} \mathscr{L}(\theta)$$

We can now write the gradient w.r.t. h_i

$$\nabla_{\mathbf{h_i}} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial h_{i1}} \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{i2}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{in}} \end{bmatrix} = \begin{bmatrix} (W_{i+1,\cdot,1})^T \nabla_{a_{i+1}} \mathscr{L}(\theta) \\ (W_{i+1,\cdot,2})^T \nabla_{a_{i+1}} \mathscr{L}(\theta) \\ \vdots \\ (W_{i+1,\cdot,n})^T \nabla_{a_{i+1}} \mathscr{L}(\theta) \end{bmatrix}$$

$$= (W_{i+1})^T (\nabla_{a_{i+1}} \mathscr{L}(\theta))$$

Here as shown above we will take the transpose of the weight matrix and we will take that dot product with the other vector matrix(gradient of the loss function the next set of the activation unit) and we will find the final value. Like that it is computed.

Therefore, for any hidden layer $\mathbf{h_i}$ we will given that as

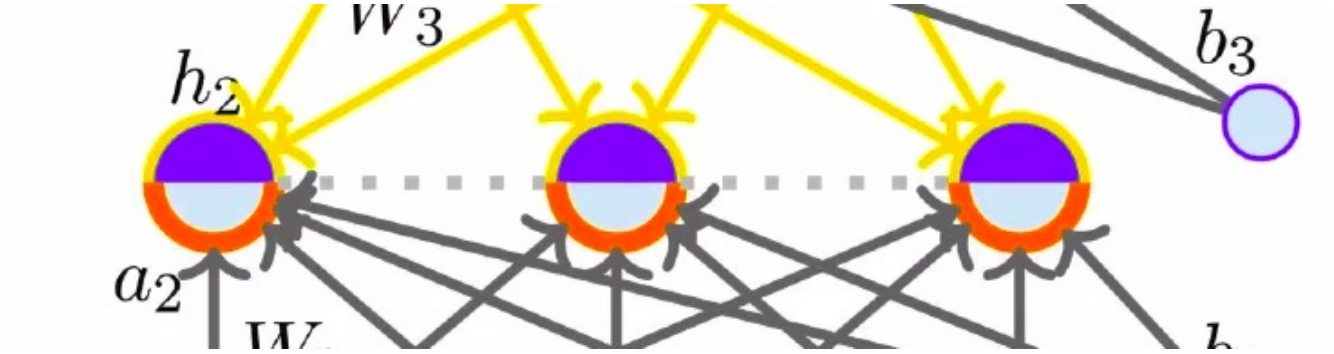
$$\nabla_{h_i} L(\theta) = (W_{i+1})^T(\nabla a_{i+1} L(\theta))$$

We have computed only for the output layer that is dark blue part and we didn't compute it for the generic lite blue part so we will compute this now.that is for the layer **i < L-1.**

- We are almost done except that we do not know how to calculate $\nabla_{a_{i+1}} \mathscr{L}(\theta)$ for $i < L-1$
- We will see how to compute that

> **Now computing the light blue regions which are below the dark blue regions.**

Here $\mathbf{a_i}$ is the light blue portion part in the third layer, the below half part.



the ones highlighted with the orange colour.

here now if we take the $\mathbf{a_i}$ part the gradient of the $\mathbf{a_i}$ with respect to the loss function is given as the follows.Its is the matrix containing the partial derivatives of the loss function w.r.t $\mathbf{a_{i1}}$ **to** $\mathbf{a_{in}}$ as shown below.

$$\nabla_{\mathbf{a_i}} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i1}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial a_{in}} \end{bmatrix}$$

Now from the above we need to compute the value of $\partial L(\theta)/\partial a_{ij}$ we will give that as below.

$$\frac{\partial \mathscr{L}(\theta)}{\partial a_{ij}} = \frac{\partial \mathscr{L}(\theta)}{\partial h_{ij}} \frac{\partial h_{ij}}{\partial a_{ij}}$$

> **We have already calculated the value of $\partial L(\theta)/\partial h_{ij}$ previously. Now we only need to calculate the $\partial h_{ij}/\partial a_{ij}$ that will be computed as the below.**

$\mathbf{a_2} = [a_{21}\ a_{22}\ a_{23}]$ and h_2 is given as

$\mathbf{h_2} = [\sigma(a_{21})\ \sigma(a_{22})\ \sigma(a_{23})]$

$\mathbf{h_2} = [h_{21}\ h_{22}\ h_{23}]$

$$\nabla_{\mathbf{a_i}} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial a_{i1}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial a_{in}} \end{bmatrix}$$

$$\frac{\partial \mathscr{L}(\theta)}{\partial a_{ij}} = \frac{\partial \mathscr{L}(\theta)}{\partial h_{ij}} \frac{\partial h_{ij}}{\partial a_{ij}}$$

$$= \frac{\partial \mathscr{L}(\theta)}{\partial h_{ij}} g^{'}(a_{ij}) \quad [\because h_{ij} = g(a_{ij})]$$

$$\nabla_{\mathbf{a_i}} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial h_{i1}} g^{'}(a_{i1}) \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{in}} g^{'}(a_{in}) \end{bmatrix}$$

$$= \nabla_{h_i} \mathscr{L}(\theta) \odot [\ldots, g^{'}(a_{ik}), \ldots]$$

Its is computed as the above like that the final result is concluded and the the product between the two vectors is not the dot product its is taking each element of the first vector and multiplying that with the corresponding elements of the second vector.

Gradient w.r.t weight and biases:

So far we have computed two parts and now we are in the third part of the computation that is finding the derivative of the loss function with respect to all the weights and biases in any particular layer.

$$\mathbf{a_k} = \mathbf{b_k} + W_k \mathbf{h_{k-1}}$$

$$\frac{\partial a_{ki}}{\partial W_{kij}} = h_{k-1,j}$$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W_{kij}} = \frac{\partial \mathscr{L}(\theta)}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial W_{kij}}$$

$$= \frac{\partial \mathscr{L}(\theta)}{\partial a_{ki}} h_{k-1,j}$$

$$\nabla_{W_k} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial W_{k11}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k12}} & \cdots & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k1n}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{knn}} \end{bmatrix}$$

Lets take a simple example of a $W_k \in \mathbb{R}^{3\times 3}$ and see what each entry looks like

$$\nabla_{W_k} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial W_{k11}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k12}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k13}} \\ \frac{\partial \mathscr{L}(\theta)}{\partial W_{k21}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k22}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k23}} \\ \frac{\partial \mathscr{L}(\theta)}{\partial W_{k31}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k32}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{k33}} \end{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial W_{kij}} = \frac{\partial \mathscr{L}(\theta)}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial W_{kij}}$$

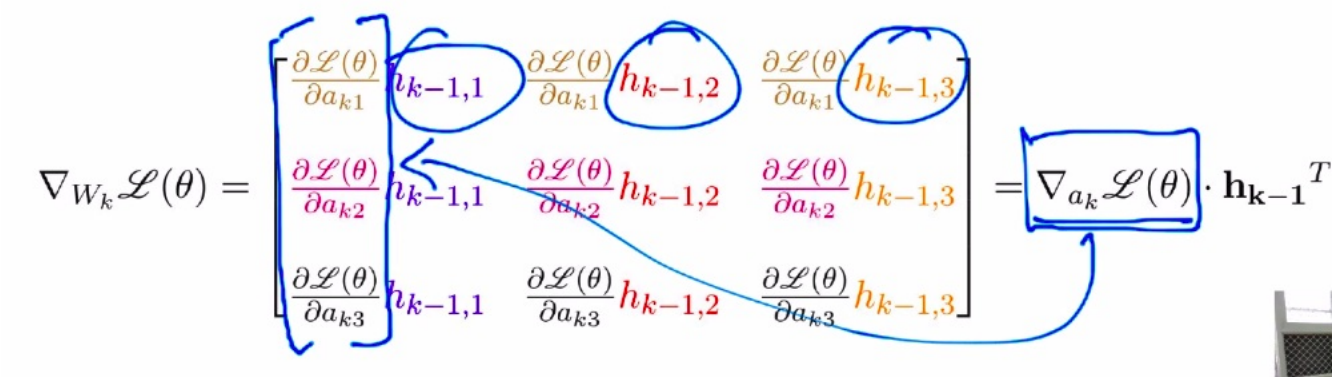
$$\nabla_{W_k} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial a_{k1}} h_{k-1,1} & \frac{\partial \mathscr{L}(\theta)}{\partial a_{k1}} h_{k-1,2} & \frac{\partial \mathscr{L}(\theta)}{\partial a_{k1}} h_{k-1,3} \\ \frac{\partial \mathscr{L}(\theta)}{\partial a_{k2}} h_{k-1,1} & \frac{\partial \mathscr{L}(\theta)}{\partial a_{k2}} h_{k-1,2} & \frac{\partial \mathscr{L}(\theta)}{\partial a_{k2}} h_{k-1,3} \\ \frac{\partial \mathscr{L}(\theta)}{\partial a_{k3}} h_{k-1,1} & \frac{\partial \mathscr{L}(\theta)}{\partial a_{k3}} h_{k-1,2} & \frac{\partial \mathscr{L}(\theta)}{\partial a_{k3}} h_{k-1,3} \end{bmatrix} =$$

Now we can write the output as the follows.

$$\nabla_{W_k}\mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial\mathscr{L}(\theta)}{\partial a_{k1}}h_{k-1,1} & \frac{\partial\mathscr{L}(\theta)}{\partial a_{k1}}h_{k-1,2} & \frac{\partial\mathscr{L}(\theta)}{\partial a_{k1}}h_{k-1,3} \\ \frac{\partial\mathscr{L}(\theta)}{\partial a_{k2}}h_{k-1,1} & \frac{\partial\mathscr{L}(\theta)}{\partial a_{k2}}h_{k-1,2} & \frac{\partial\mathscr{L}(\theta)}{\partial a_{k2}}h_{k-1,3} \\ \frac{\partial\mathscr{L}(\theta)}{\partial a_{k3}}h_{k-1,1} & \frac{\partial\mathscr{L}(\theta)}{\partial a_{k3}}h_{k-1,2} & \frac{\partial\mathscr{L}(\theta)}{\partial a_{k3}}h_{k-1,3} \end{bmatrix} = \nabla_{a_k}\mathscr{L}(\theta)\cdot \mathbf{h_{k-1}}^T$$

here we came to a conclusion like this the final result is given as the product like below.

The first term in the final expression(the gradient of loss function) is the particular all the partial derivatives common all the three columns it is the collection all the partial derivatives.



and $\mathbf{h_{k-1}^T}$ is the $\mathbf{h_{k-1,1}, h_{k-1,2}, h_{k-1,3}}$ the circled portion in the above picture

$\mathbf{h_{k-1,1}, h_{k-1,2}, h_{k-1,3}}$ are the activations of the layer k-1 so it can we written as single layer $\mathbf{h_{k-1}^T}$ here T is the transpose **that is because this vector need to be sleeping vector(Horizontal) and the gradient needs to be the stand vector(Vertical) while the taking the dot product**.

Finally we found the derivatives with respect to weights W those all can eb computed in one go that is

$W_k = W_k - \eta$***matrix $[\nabla w_i L(\theta)]$**

> $W_k = W_k - \eta^*[\nabla a_k L(\theta).\ h_{k-1}^T]$

Finally, we are only left with biases, we need to calculate the derivative of the loss function w.r.t biases.

We have bias corresponding to every neuron in the layer.

$$a_{ki} = b_{ki} + \sum_j W_{kij} h_{k-1,j}$$

$$\frac{\partial \mathscr{L}(\theta)}{\partial b_{ki}} = \frac{\partial \mathscr{L}(\theta)}{\partial a_{ki}} \frac{\partial a_{ki}}{\partial b_{ki}}$$

$$= \frac{\partial \mathscr{L}(\theta)}{\partial a_{ki}}$$

Here the **$\partial L(\theta)/\partial a_{ki}$** is the **partial derivative of the loss function w.r.t to i^{th} bias in the k^{th} layer.**

We can now write the gradient w.r.t. the vector b_k

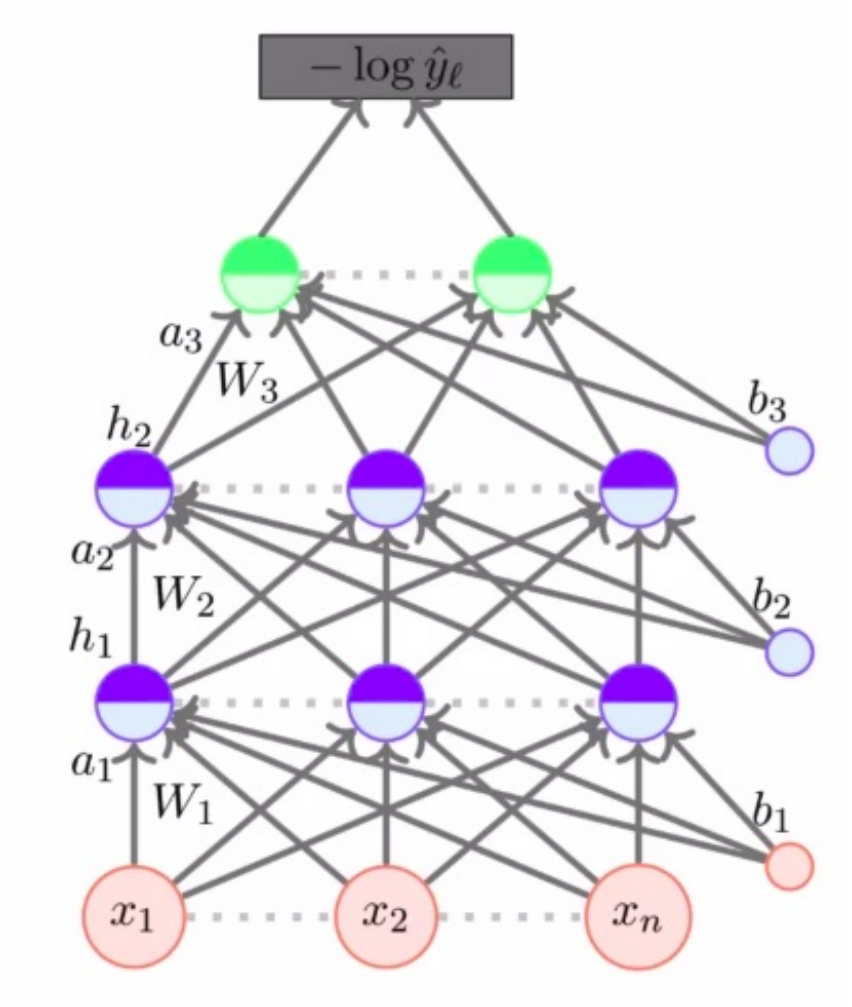
$$\nabla_{\mathbf{b_k}} \mathscr{L}(\theta) = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{a_{k1}} \\ \frac{\partial \mathscr{L}(\theta)}{a_{k2}} \\ \vdots \\ \frac{\partial \mathscr{L}(\theta)}{a_{kn}} \end{bmatrix} = \nabla_{\mathbf{a_k}} \mathscr{L}(\theta)$$

Here i is replaced with all the vertices numbers from 1 to n. It is nothing but all the partial derivatives of the loss function w.r.t the layer a_k

Now we can write the **b_k** finally as

> $b_k = b_k - \eta \nabla {}^{b_k} L(\theta)$

Finally we have completed all the stages and finally we are at the learning algorithm part. All put together until now.



This the basic example we take and the as discuss the learning algorithm

we will initialise the weights randomly and we will go through forward pass of the data.

Forward Pass:

Algorithm: forward_propagation(θ)

for $k = 1$ *to* $L - 1$ **do**

$a_k = b_k + W_k h_{k-1}$;

$h_k = g(a_k)$;

end

$a_L = b_L + W_L h_{L-1}$;

$\hat{y} = O(a_L)$;

In the **forward pass**(Algorithm Explanation):

1. Taking fro loop we iterate from first layer to the L-1 layer that is just before the hidden layer.
2. Compute the pre-activation $a_k = b_k - W_k h_{k-1}$
3. Compute the activation applying the non linearity (it can be sigmoid function $h_k = g(a_k)$
4. End the for loop using this we will compute all except the last layer.

> **Now we need to compute the last layer.**

5.here the light green parts will be equal to the weighted sum of the inputs that is **$a_i = n + W_i h_{i-1}$**

6.Finally the output **$\hat{y}$** is some function applied to light green parts we see in this case the function we apply is the softmax function.

Like this we have computed all the **$a_1, h_1, a_2, h_2, a_3, h_3$** and **both the final y's** in the last layer and finally the **$\hat{y}$.**

> **This is known as the forward pass or the forward propagation.It is very simple numpy coded.**

After the forward pass we need to propagate back towards all the weights in the network.

Back-Propagation:

Algorithm: back_propagation$(h_1, h_2, ..., h_{L-1}, a_1, a_2, ..., a_L, y, \hat{y})$

//Compute output gradient ;
$\nabla_{a_L} \mathscr{L}(\theta) = -(y - \hat{y})$;
for $k = L$ *to* 1 **do**
    // Compute gradients w.r.t. parameters ;
    $\nabla_{W_k} \mathscr{L}(\theta) = \nabla_{a_k} \mathscr{L}(\theta) h_{k-1}^T$;
    $\nabla_{b_k} \mathscr{L}(\theta) = \nabla_{a_k} \mathscr{L}(\theta)$;
    // Compute gradients w.r.t. layer below ;
    $\nabla_{h_{k-1}} \mathscr{L}(\theta) = W_k^T (\nabla_{a_k} \mathscr{L}(\theta))$;
    // Compute gradients w.r.t. layer below (pre-activation);
    $\nabla_{a_{k-1}} \mathscr{L}(\theta) = \nabla_{h_{k-1}} \mathscr{L}(\theta) \odot [\ldots, g'(a_{k-1,j}), \ldots]$;
end

1. we will compute the gradient with the formula of -(y-ŷ).
2. Taking a for loop iterate fro the last layer to the first layer i.e., form L to 1

$$\nabla_{W_k} \mathscr{L}(\theta) = \nabla_{a_k} \mathscr{L}(\theta) h_{k-1}^T \; ;$$
$$\nabla_{b_k} \mathscr{L}(\theta) = \nabla_{a_k} \mathscr{L}(\theta) \; ;$$

3. Compute the gradients w.r.t the w and b in beginning of each iteration as shown in the above picture.

4. Now computing the derivative w.r.t to the dark blue part.

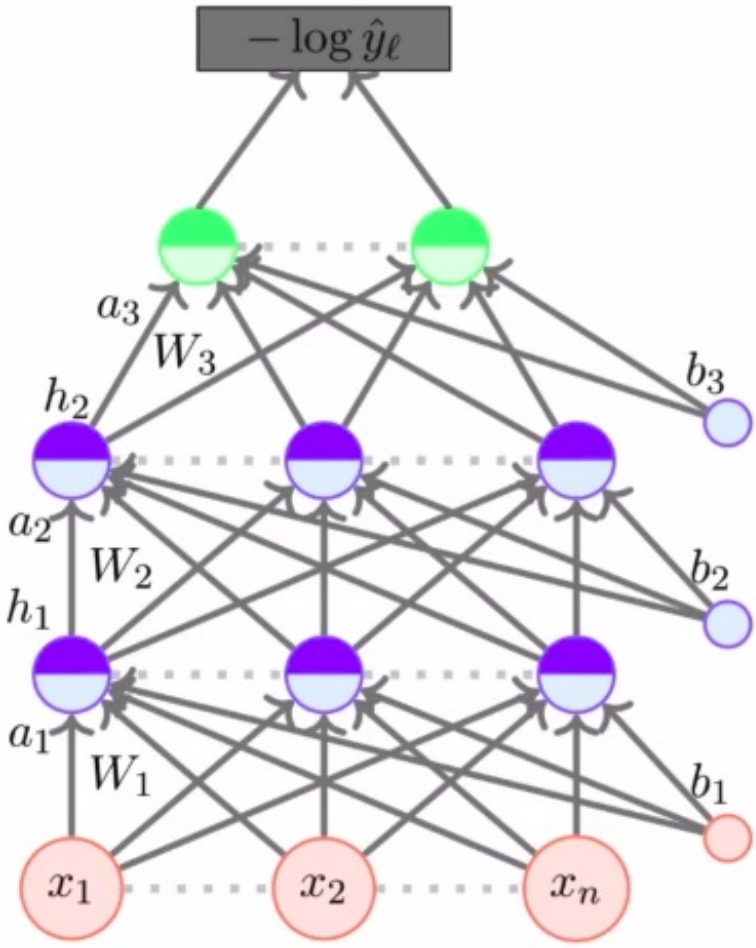
5. Compute the gradients w.r.t layers below which is the pre-activation layer.

6. End the for loop.

Like that we will do the back Propagation.

This is about the back propagation, all theory an the learning algorithm part.

Example on the back propagation with a sample values:



This is is the basic network you take and then we have following **w and b as below ones**.

$$b = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} 0.1 & 0.3 & 0.8 \\ -0.3 & -0.2 & 0.5 \\ -0.3 & 0.1 & 0.4 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.4 & 0.5 & -0.3 \\ -0.1 & -0.4 & -0.5 \\ 0.8 & 0.2 & 0.9 \end{bmatrix} \quad W_3 = \begin{bmatrix} 0.3 & -0.5 \\ 0.1 & 0.2 \\ -0.1 & -0.4 \end{bmatrix}$$

W and b

These are the w and b of the the neural network. Now we randomly assign some values and we have the true outputs:

$$x = \begin{bmatrix} 2 & 5 & 3 \end{bmatrix} \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

randomly chosen x and the true outputs y of the network

With the random values of the x and the all the values which we have we will perform the forward pass with help of the algorithm we have and we arrive at some particular predicted values as shown below.

Output :

$$a_1 = W_1 * x + b_1 = [\ -2.2 \quad -0.1 \quad 5.3\ ]$$

$$h_1 = tanh(a_1) = [\ -0.97 \quad -0.1 \quad 0.99\ ]$$

$$a_2 = W_2 * h_1 + b_2 = [\ 0.41 \quad -0.24 \quad 1.24\ ]$$

$$h_2 = tanh(a_2) = [\ 0.39 \quad -0.24 \quad 0.84\ ]$$

$$a_3 = W_3 * h_2 + b_3 = [\ 0.01 \quad -0.58\ ]$$

$$\hat{y} = softmax(a_3) = [\ 0.64 \quad 0.36\ ]$$

$$L(\Theta) = -\log \hat{y}_l = 0.43$$

The loss which we got was 0.43 as the true and predicted values differed.So far we are done with the forward pass and got the loss as 0.43 now we will back propagate with help of the back propagation algorithm.

$$\nabla_{a_L} \mathscr{L}(\theta):$$

$$= \begin{bmatrix} \frac{\partial \mathscr{L}\theta}{\partial a_{L1}} \\ \\ \frac{\partial \mathscr{L}\theta}{\partial a_{L2}} \end{bmatrix} = \begin{bmatrix} -(1_{\ell=1} - \hat{y}_1) \\ \\ -(1_{\ell=2} - \hat{y}_2) \end{bmatrix}$$

$$= \begin{bmatrix} -(1 - 0.64) \\ \\ -(0 - 0.36) \end{bmatrix}$$

$$= \begin{bmatrix} -0.36 \\ \\ 0.36 \end{bmatrix}$$

Computing the gradient.(Step1 in the algorithm)

With this after step one we will go will go to update the gradient w.r.t layer below.

$$W_3^T = \begin{bmatrix} 0.3 & -0.5 \\ 0.1 & 0.2 \\ -0.1 & -0.4 \end{bmatrix}$$

$$\nabla_{h_2} \mathscr{L}(\theta):$$

$$= \begin{bmatrix} \frac{\partial \mathscr{L}\theta}{\partial h_{21}} \\ \frac{\partial \mathscr{L}\theta}{\partial h_{22}} \\ \frac{\partial \mathscr{L}\theta}{\partial h_{23}} \end{bmatrix} \quad = W_3^T \nabla_{a_3} \mathscr{L}(\theta)$$

$$\nabla_{h_2} \mathscr{L}(\theta):$$

$$= \begin{bmatrix} \frac{\partial \mathscr{L}\theta}{\partial h_{21}} \\ \frac{\partial \mathscr{L}\theta}{\partial h_{22}} \\ \frac{\partial \mathscr{L}\theta}{\partial h_{23}} \end{bmatrix} = W_3^T \nabla_{a_3} \mathscr{L}(\theta)$$

$$= \begin{bmatrix} 0.23 \\ -0.03 \\ 0.09 \end{bmatrix}$$

Now we will come to compute the derivative of loss function with pre-activation layer that is a_2:

$$\nabla_{a_2} \mathscr{L}(\theta):$$

$$= \begin{bmatrix} \frac{\partial \mathscr{L}\theta}{\partial a_{21}} \\ \frac{\partial \mathscr{L}\theta}{\partial a_{22}} \\ \frac{\partial \mathscr{L}\theta}{\partial a_{23}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathscr{L}(\theta)}{\partial h_{21}} g'(a_{21}) \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{22}} g'(a_{22}) \\ \frac{\partial \mathscr{L}(\theta)}{\partial h_{23}} g'(a_{23}) \end{bmatrix}$$

$$= \begin{bmatrix} 0.23 * 0.84 \\ -0.03 * 0.96 \\ 0.09 * 0.28 \end{bmatrix} = \begin{bmatrix} 0.19 \\ -0.03 \\ 0.02 \end{bmatrix}$$

Now we come to the updating of w and b :

$$\nabla_{W_2}\mathcal{L}(\theta):$$

$$= \begin{bmatrix} \frac{\partial \mathcal{L}\theta}{\partial a_{21}} h_{11} & \frac{\partial \mathcal{L}\theta}{\partial a_{21}} h_{12} & \frac{\partial \mathcal{L}\theta}{\partial a_{21}} h_{13} \\ \frac{\partial \mathcal{L}\theta}{\partial a_{22}} h_{11} & \frac{\partial \mathcal{L}\theta}{\partial a_{22}} h_{12} & \frac{\partial \mathcal{L}\theta}{\partial a_{22}} h_{13} \\ \frac{\partial \mathcal{L}\theta}{\partial a_{23}} h_{11} & \frac{\partial \mathcal{L}\theta}{\partial a_{23}} h_{12} & \frac{\partial \mathcal{L}\theta}{\partial a_{23}} h_{13} \end{bmatrix}$$

$$= \begin{bmatrix} -0.19 & 0.03 & -0.02 \\ -0.02 & 0.01 & -0.01 \\ 0.2 & -0.03 & 0.02 \end{bmatrix}$$

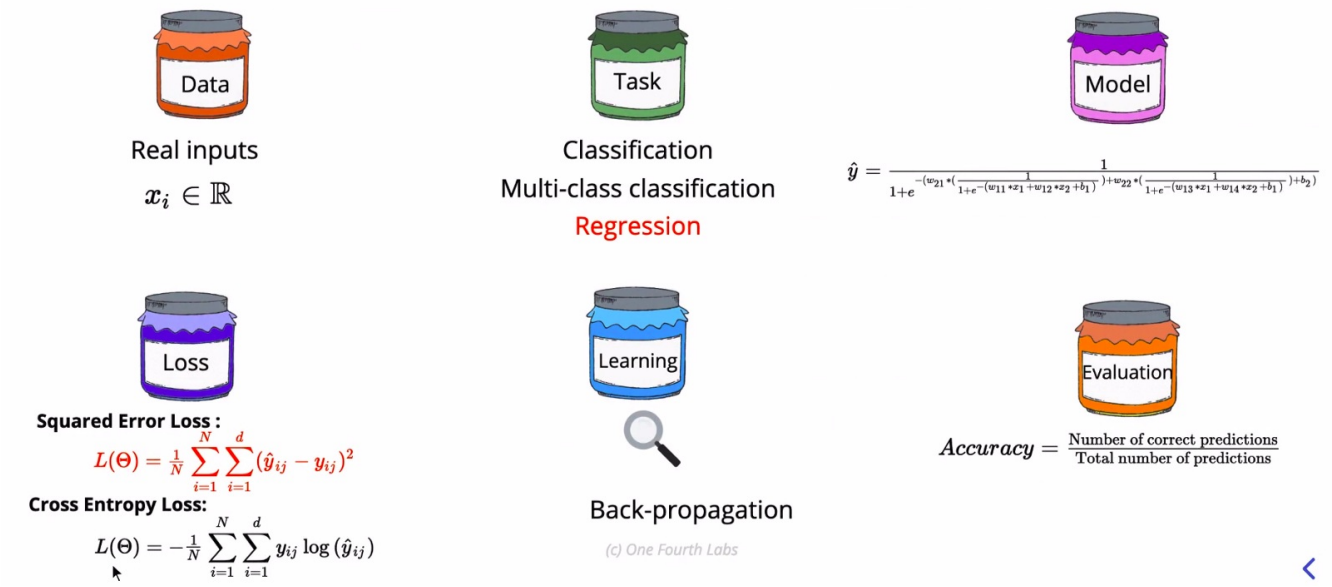
Now the updating od w is like

$w_2 = w_2 - \eta * \nabla\, w_2\, L(\theta)$

In the same way we will update the bias like this.

This how the back propagation works which is explained with an example.

Summary:



This all about the back propagation part of the course the heavy maths part behind implementing the back propagation.

This is a small try ,uploading the notes . I believe in "**Sharing knowledge is that best way of developing skills**".Comments will be appreciated. Even small edits can be suggested.

> **Each Applause will be a great encouragement. Feel free to suggest and Comment.**

Do follow my medium for more updates.....